A Batchwise Monotone Algorithm for Dictionary Learning

Huan Wang
Yahoo!Labs, New York

John Wright
Computer Science Department, Columbia University,

Daniel Spielman
Computer Science Department, Yale University

February 3, 2015

Abstract

We propose a batchwise monotone algorithm for dictionary learning. Unlike the state-of-the-art dictionary learning algorithms which impose sparsity constraints on a sample-by-sample basis, we instead treat the samples as a batch, and impose the sparsity constraint on the whole. The benefit of batchwise optimization is that the non-zeros can be better allocated across the samples, leading to a better approximation of the whole. To accomplish this, we propose procedures to switch non-zeros in both rows and columns in the support of the coefficient matrix to reduce the reconstruction error. We prove in the proposed support switching procedure the objective of the algorithm, i.e., the reconstruction error, decreases monotonically and converges. Furthermore, we introduce a block orthogonal matching pursuit algorithm that also operates on sample batches to provide a warm start. Experiments on both natural image patches and UCI data sets show that the proposed algorithm produces a better approximation with the same sparsity levels compared to the state-of-the-art algorithms.

1 Introduction

A number of algorithms have recently been developed that automatically design representations through a process called dictionary learning. The hope is that learning algorithms can exploit structure in specific classes of signals, enabling better performance in applications. Dictionary learning algorithms have already been used successfully in a number of image processing problems, such as image compression [8][6], inpainting [10][13], image denoising [8][16][2][12], super-resolution [5][4], digit recognition, and texture classification [11].

Popular dictionary learning algorithms can be roughly divided into two categories: hard constraint-based [8] [3], and soft sparsity-penalty-based [10][1][7]. These algorithms search for a dictionary of vectors (called atoms) so that it is possible to represent each sample signal as a linear combination of a small number of the atoms. Often dictionaries with more atoms than the dimension, called over-complete dictionaries, are used.

Since we would like to use only a few atoms in the representation of each sample, a sparsity constraint is imposed on the coefficients in the representations. Both K-SVD [8] and the online dictionary learning [10] algorithm impose sparsity constraints, either hard or soft, on the representation of individual samples. However, it may not be optimal to assume a similar sparsity level for each sample. In fact, some samples could be easy to represent and some may require more atoms in their representations. The recent dictionary learning algorithm of [17] searches for dictionaries that have a sparsity constraint on the number of times each atom is used. Thus, some signals can be represented using more atoms than others. Their algorithm inspires us to focus on how individual atoms are used rather than how individual signals are represented.

In this paper we present a monotone algorithm for dictionary learning. Similar to [17], the algorithm we propose in this work also acts on the rows of the coefficient matrix, but can empirically produce good approximations even in the more challenging (and realistic) conditions. In contrast to the traditional sample-based sparsity constraint, we impose the sparsity constraint in a batchwise fashion. That is, we switch the positions of the non-zeros in the coefficients within batches of samples among different columns and rows in the coefficient matrix, and at the same time keep the total number of non-zeros fixed. As a result, the number of non-zeros, constrained within a batch of samples, is allowed to vary in either columns or rows . We show that all the non-zero position switching operations only reduce reconstruction error, leading to a convergent objective function. For initialization, we introduce a simple iterative dictionary update procedure that operates on a batch of samples to give an approximate guess of the dictionary. In each iteration, first the non-zero patterns are derived using a block orthogonal matching pursuit and then the dictionary is updated using least squares.

There are two main advantages of our proposed algorithm:

1. Since the non-zero positions are optimized in a batchwise fashion, we are able to achieve a smaller reconstruction error, or better approximation, compared to the traditional sample-by-sample constraint with the same level of sparsity.

2. The reconstruction error is guaranteed to decrease monotonically and converge.

2 Notation

In the dictionary learning problem, one is given a matrix that contains the sample signals in its columns, $Y = [y_1, y_2, \ldots, y_p] \in \mathbb{R}^{m \times p}$, along with a target number of atoms, n. The goal is to find a dictionary of atoms $A \in \mathbb{R}^{m \times n}$ and a sparse coefficient matrix $X \in \mathbb{R}^{n \times p}$ so that $Y \approx AX$.

Throughout this paper, m is the dimension of each sample and p is the number of samples. We use a_i and x_i to denote the ith column of A and X respectively, and x^i

to denote the ith row of X. We use Ω^i to denote the support of x^i, Ω_i for the support of x_i, $k^i = |\Omega^i|$, and $k_i = |\Omega_i|$. $\otimes$ is the Kronecker product, I_p is a p-by-p identity matrix, and $y = \mathrm{vec}(Y)$, where $\mathrm{vec}(\cdot)$ concatenates the columns of Y to form a vector.

For a set $\Omega \subseteq \{1, \ldots, \}$, we let P_Ω to denote the projection matrix $P_\Omega = [e_{\omega_1}, e_{\omega_2}, \ldots, e_{\omega_{|\Omega|}}]$, where e_i is the elementary unit vector in coordinate i. We use Y_Ω to denote YP_Ω. A/a_i means the sub-matrix constructed by removing the ith column of A, and X/x^i is the sub-matrix constructed by removing the ith row of X. When describing iterative algorithms, we use $x^{(+)}$ to denote the updated value of x.

3 Columnwise Sparsity Constraints

State-of-the-art dictionary learning algorithms treat the input sample matrix Y in a sample-by-sample way. That is, the sparseness constraint is imposed on the coefficient of each sample independently using the current dictionary, and then the dictionary and coefficient matrix are updated accordingly. Among popular dictionary learning algorithms, two representative ones are the hard-constraint-based K-SVD [8] and the soft-penalty-based dictionary learning algorithms such as the online learning algorithm [10] and the efficient sparse coding algorithms [7].

The K-SVD algorithm aims to iteratively minimize the objective

$$\min_{A,X} \|Y - AX\|_f^2 \text{ s.t. } \forall i, \|x_i\|_0 \leq k, \tag{1}$$

where A is the dictionary and X is the sparse coefficient matrix.

Empirically the K-SVD algorithms often works well, but its objective value is not guaranteed to decrease monotonically because the support of X is changed using the greedy pursuit algorithm one column at a time.

The algorithms in [10] [7] replace the hard ℓ_0 penalty with an ℓ_1 penalty, giving

$$\min_{A,X} \|Y - AX\|_f^2 + \lambda \sum_i \|x_i\|_1. \tag{2}$$

This optimization problem is not convex. However, it is convex in X if A is fixed, or in A if X is fixed. As in the method of optimal directions (MOD), the optimization is done via alternating directions.

The advantage of column-wise sparsity constraints is that it leads naturally to fast online algorithms. Whenever a new sample comes one simply adds a sparsity constraint on the incoming column of X. The downside is the sample-by-sample sparsity treatment lacks a "global" view of the sparsity pattern. For example, there is no reason we should require each sample to be represented by exactly k atoms in the dictionary, or impose a sparsity penalty with the same λ. Some samples, being "harder" to approximate, require more atoms, and it could be a waste to use too many atoms to represent the "easy" samples.

Algorithm 1 Inner Row Support Switching

Input: $\tilde{Y} \in \mathbb{R}^{m\times p}$, $x \in \mathbb{R}^{1\times p}$, $a \in \mathbb{R}^{m\times 1}$, and N.
Output: $a^{(+)} \in \mathbb{R}^{m\times 1}$, and $x^{(+)} \in \mathbb{R}^{1\times p}$.
Denote the support of x as Ω, and $k = |\Omega|$.
for $i = 1$ **to** N **do**
  SVD: $\tilde{Y}_\Omega = \sigma_1 a x_\Omega + \sum_{1<j\leq m} \sigma_j u_j v_j^T$, s.t., $\sigma_1 \geq \sigma_2 \cdots \geq \sigma_m$, and $\|a\|_2 = 1$.
  Use the indices of the k largest $\{|a^T \tilde{y}_i|\}$ as Ω.
  $x_\Omega = a^T \tilde{Y}_\Omega / \|a\|_2^2$, $x_{\Omega^c} = 0$.
end for
$a^{(+)} = a$, $x^{(+)} = x$.

4 Batchwise Support Switching Procedures

Unlike K-SVD and online learning, which constrain the sparsity of the coefficients in a column-by-column fashion, we argue that it may be possible to obtain better sparse approximations of the input as a whole if we allow the column sparsity to vary. We seek the best possible reconstruction, subject to a constraint on the *total* number of nonzeros across the batch of samples:

$$\min_{A,X} \|Y - AX\|_f^2 \text{ s.t. } \|X\|_0 \leq K, \tag{3}$$

The advantage is that some non-zero positions with less impact on the objective can be replaced using crucial ones. As a result a more accurate decomposition is produced with different column sparsities across samples.

We introduce a heuristic for attacking this problem, which computes an initial sparsifying dictionary using alternating directions, and then refines it using sequence of support and amplitude adjustments. The initial approximation makes use of a *batch-wise* orthogonal matching pursuit, which aims at minimizing the ℓ_0 norm of X as a whole.

The support switching procedure updates the non-zero positions, i.e., the sparsity patterns, in the coefficient matrix X in two ways: inner-row switching and inter-row switching. In the inner-row switching, the total number of non-zeros in each row is fixed, and the non-zero positions are adjusted within the same row; and in the inter-row switching, the total number of non-zeros in pairs of rows is fixed, and the non-zeros are changed between two rows. Finally, we introduce an iterative procedure to adjust the amplitude of the coefficient and dictionary, with the sparsity pattern fixed. The whole algorithm is described in Algorithm 4. We prove that in the procedure of sparsity pattern switching and the amplitude adjustment, the objective decreases monotonically and converges.

Algorithm 2 Inter Row Support Switching

Input: $\tilde{Y} \in \mathbb{R}^{m\times p}$, $x^i, x^j \in \mathbb{R}^{1\times p}$, and $a_i, a_j \in \mathbb{R}^{m\times 1}$.
Output: $x^{i(+)}, x^{j(+)} \in \mathbb{R}^{1\times p}$.
1. Denote the support of x^i and x^j as Ω^i, and Ω^j respectively, and $\Omega = (\Omega^i \cup \Omega^j) \setminus (\Omega^i \cap \Omega^j)$. Denote $\tilde{\Omega} = (\Omega_i \cap \Omega_j)^{\mathsf{c}}$.
2. Form the matrix $M = [a_i, a_j]^T \tilde{Y}_{\tilde{\Omega}}$.
3. Pick up the larger entry of the two rows in each column in $|M|$ as candidates, and use the positions of the largest $|\Omega|$ candidates as $\Omega^{(+)}$.
4. Set $X_{\Omega^{(+)}} = M_{\Omega^{(+)}}$, $X_{\tilde{\Omega}\setminus\Omega^{(+)}} = 0$, and $X_{\Omega_i\cap\Omega_j} = [x^i; x^j]_{\Omega_i\cap\Omega_j}$.
5. Return $x^{i(+)} = e_1^T X$ and $x^{j(+)} = e_2^T X$.

4.1 Inner-Row Support Switching

Suppose we are given the number of non-zeros k^i in each row of X. The problem becomes

$$\min_{A,X} \|Y - AX\|_f^2 \quad \text{s.t.} \quad \|x^i\|_0 = k^i, \quad i = 1, \ldots, p. \tag{4}$$

Note here x^i is the i-th row of X compared to the i-th column in the K-SVD objective. Globally optimizing this objective is challenging, due to the nonconvexity of the constraint set and the objective. Similar to K-SVD, we can obtain a simpler subproblem by only considering one row at a time, giving

$$\min_{a_i, x^i} \left\|Y - \sum_{j\neq i} a_j x^j - a_i x^i\right\|_f^2 \quad \text{s.t.} \quad \|x^i\|_0 \leq k^i. \tag{5}$$

Setting $\tilde{Y} = Y - \sum_{j\neq i} a_j x^j$, the problem becomes one of finding a rank-one approximation to $\tilde{Y}$, with at most k^i nonzero columns.

We attack this problem using alternating directions. Assuming the support Ω^i is known and fixed, a best rank one approximation $a_i x^i$ can be fit using the SVD of $\tilde{Y}_{\Omega^i}$. If, on the other hand a_i is fixed, an optimal support Ω^i can be derived by simply ranking the absolute values of the projected samples, i.e., $|a_i^T y_i|$. Once the support Ω^i and the atom a_i are known, x^i can be calculated in closed-form. The resulting algorithm is listed in Algorithm 1.

4.2 Inter-Row Support Switching

In this section, we introduce a procedure to adjust the non-zeros between two rows of X, such that the reconstruction error in (4) decreases and at the same the total number of non-zeros in the two rows stays the same. First, we define the unique columns in $X_s = [x^i; x^j]$ to be the symmetric difference of the supports of x^i and x^j:

Definition Suppose the support of x^i and x^j are Ω^i and Ω^j respectively, then the index set of the unique columns in $\begin{bmatrix} x^i \\ x^j \end{bmatrix}$ are $\Omega = (\Omega^i \cup \Omega^j) \setminus (\Omega^i \cap \Omega^j)$.

If we fix the remaining columns, the residual is $\tilde{Y} = Y - [A \backslash \{a_i, a_j\}][X \backslash \{x^i, x^j\}]$. Again, if we fix the dictionary atoms a_i and a_j, if $\|a_i\| = \|a_j\| = 1$, the optimal support for the $|\Omega|$ unique columns can be derived by ranking the absolute values of the projected sample $M = [a_i, a_j]^T \tilde{Y}_{(\Omega_i \cap \Omega_j)^c}$ with the constraint that we can only pick up one non-zero in each column of M. The procedure is described in Algorithm 2.

The inter-row support switching reduces the objective in (4) by fixing the dictionary and comparing the importance of non-zero positions by ranking the projected absolute values of the residuals $\tilde{Y}$. If we run the procedure for all pairs of rows in X, the total number of non-zeros in X stays the same but their distribution is optimized batchwisely. The procedure is based on fixed dictionary A, and the optimization is only carried out on rows of X. Switching supports between all pairs of rows can be expensive when the number of rows n in X grows large. In application, we use two ways to reduce the computation cost:

1. Instead of going over all pairs of rows in X, we only go through a randomly sampled subset of the $\binom{n}{2}$ pairs.

2. The inter-row support switching is only carried out when the objective decreases very slowly in the inner-row support switching.

The inner-row and inter-row support switching interchange the positions of the non-zeros in the coefficient matrix within a batch of samples. In the following section we will introduce a procedure to further reduce the objective by changing the amplitude of the entries in both A and X given the support Ω of X.

5 Alternating Amplitude Adjustment

Algorithm 3 Alternating Amplitude Adjustment

Input: $Y \in \mathbb{R}^{m \times p}$, $X \in \mathbb{R}^{n \times p}$, $A \in \mathbb{R}^{m \times n}$, and N.
Output: $X^{(+)} \in \mathbb{R}^{n \times p}$, and $A^{(+)} \in \mathbb{R}^{m \times n}$.
Denote the support of X as Ω, and the support of x_i as Ω_i.
for $i = 1$ **to** N **do**
  $A = YX^T(XX^T)^{-1}$.
  for $j = 1$ **to** p **do**
    $x_j(\Omega_j) = (A_{\Omega_j}^T A_{\Omega_j})^{-1} A_{\Omega_j}^T y_j$
    $x_j(\Omega_j^c) = 0$
  end for
end for
$X^{(+)} = X$, $A^{(+)} = A$.

In this section, we propose an alternating optimization algorithm for the following problem:

$$\min_{A,X} \ \|Y - AX\|_f^2, \quad \text{s.t.} \ \ X(\Omega^c) = 0 \tag{6}$$

Algorithm 4 BatchSVD

Input: $Y \in \mathbb{R}^{m\times p}$, $A \in \mathbb{R}^{m\times n}$,$X \in \mathbb{R}^{n\times p}$, N_1, ϵ, N_2.
Output: $A^{(+)} \in \mathbb{R}^{m\times n}$, and $X^{(+)} \in \mathbb{R}^{n\times p}$.
Rearrange column of A and rows of X such that such that $k^1 \geq k^2 \geq \cdots \geq k^n$.
repeat
  for $i = 1$ **to** N_1 **do**
    Run Algorithm 1 with input $\tilde{Y} = Y - [A \setminus a_i][X \setminus x^j]$, x^i, a_i, and N. Update x^i, and a_i.
  end for
  Rescale A and X, such that $\forall i, \|A_i\|_2 = 1$.
  for all $\binom{n}{2}$ **pairs of rows** {i,j} **do**
    Run Algorithm 2 with input $\tilde{Y} = Y - [A\setminus\{a_i, a_j\}][X\setminus\{x^i, x^j\}]$, x^i, x^j, a_i, a_j. Update x^i and x^j.
  end for
  Run Algorithm 3 with input Y, A, X, and N_2. Update $A^{(+)}$ and $X^{(+)}$.
until $\|Y - AX\|_2^2 - \|Y - A^{(+)}X^{(+)}\|_2^2 \leq \epsilon$

If X is known and fixed, the optimal A can be computed via least squares: $A = YX^T(XX^T)^{-1}$.

On the other hand, given A and Ω, the objective above decomposes into a sum of samplewise reconstruction errors: $\min_{X_\Omega} \|Y - AX\|_f^2 = \min_{x_j(\Omega_j)} \sum_j \|y_j - A_{\Omega_j} x_j(\Omega_j)\|_2^2$, which amounts to solving least squares for each column of X sample-by-sample, that is: $x_j(\Omega_j) = (A_{\Omega_j}^T A_{\Omega_j})^{-1} A_{\Omega_j}^T y_j, \text{ and } x_j(\Omega_j^c) = 0$. The detailed procedure is shown in Algorithm 3. It is not hard to see that in each iteration the objective does not increase.

6 Proof of Monotonicity

The full procedure is described in Algorithm 4. [1]. We will show in this section that all the procedures introduced in section (4) and section (5) only decrease the objective value, while keeping the total number of nonzero coefficients unchanged.

Lemma 6.1. *The objective (4) decreases monotonically in Algorithm 1.*

Proof. Let $L(a_i, x^i, \Omega^i) = \|\tilde{Y}_i - a_i x^i(\Omega^i)\|_f^2$, where $\tilde{Y}_i = Y - \sum_{j\neq i} a_j x^j$. For monotonicity, it suffices to show $L(a_i^{(+)}, x^{i(+)}, \Omega^{i(+)}) \leq L(a_i, x^i, \Omega^i)$.

Since $L(a_i, x^i, \Omega^i) = \|\tilde{Y}_i - a_i x^i(\Omega^i)\|_f^2 = \|\tilde{Y}_{\Omega^i} - a_i x^i_{\Omega^i}\|_f^2 + \|\tilde{Y}_{\Omega^{ic}}\|_f^2$,the second term $\|\tilde{Y}_{\Omega^{ic}}\|_f^2$ is fixed given Ω^i. Since $\{a_i^{(+)}, x^{i(+)}\}$ minimizes $\|\tilde{Y}_{\Omega^i} - a_i x^i_{\Omega^i}\|_f^2$ for any given Ω^i, we have $L(a_i^{(+)}, x^{i(+)}, \Omega^i) \leq L(a_i, x^i, \Omega^i)$.

[1] Ways to reduce the computation cost is discussed in section 4.2

Algorithm 5 Dictionary Approximation using Block OMP

Input: $Y \in \mathbb{R}^{m\times p}$, A_o, N, T.
Output: $A \in \mathbb{R}^{m\times n}$ and $X \in \mathbb{R}^{n\times p}$.
Initialize $X = 0$, and $A = A_o$.
for $t = 1$ **to** T **do**
(1) $X \leftarrow$ OMP $(\mathrm{vec}(Y), I_p \otimes A, N)$.
(2) $A \leftarrow \arg\min_A \|Y - AX\|_f^2$.
end for

In the second step $a_i^{(+)}$ is fixed, and w.o.l.g. let us assume $\|a_i^{(+)}\|_2 = 1$. We would like to

$$\min_{\Omega^i, x^i} \|\tilde{Y} - a_i^{(+)} x^i\|_f^2 = \min_{\Omega^i} \min_{x^i} \sum_{j\in\Omega^i} \|\tilde{y_j} - a_i^{(+)} x^i(j)\|_2^2 + \sum_{j\notin\Omega^i} \|\tilde{y_j}\|_2^2$$

$$= \min_{\Omega^i} \sum_{j\in\Omega^i} \min_{x^i_{\Omega_i}} \|\tilde{y_j} - a_i^{(+)} x^i(j)\|_2^2 + \sum_{j\notin\Omega^i} \|\tilde{y_j}\|_2^2 = \min_{\Omega^i} \sum_{j\in\Omega^i} (\|\tilde{y_j}\|_2^2 - (a_i^{(+)T}\tilde{y_j})^2) + \sum_{j\notin\Omega^i} \|\tilde{y_j}\|_2^2$$

$$= \sum_j \|\tilde{y_j}\|_2^2 - \max_{\Omega^i} \sum_{j\in\Omega^i} (a_i^{(+)T}\tilde{y_j})^2 = \|\tilde{Y}\|_f^2 - \max_{\Omega^i} \|a_i^{(+)T}\tilde{Y}_{\Omega^i}\|_2^2 \quad (7)$$

If we would like to choose k^i non-zeros in the ith row of X, then the optimal way of minimizing the objective is to choose the ones with the largest $|a_i^T\tilde{y_j}|$. Thus Ω^i corresponding to the k^i entries with the largest $|a_i^T\tilde{y_j}|$s minimizes the reconstruction error. And the corresponding x^i is determined by the projection of $\tilde{Y}_{\Omega^j}$ onto $a_i^{(+)}$. □

In a similar way, we prove the objective (4) decreases monotonically in Algorithm 2. The proof is omitted here.

Convergence of the Objective: Since the objective values generated by the algorithm is a monotonically decreasing sequence of non-negative real numbers, we know it converges according to the monotone convergence theorem.

7 Dictionary Initialization

The proposed algorithm, though has a convergent objective, is still a local one. A natural question is how we should initialize the sparsity pattern of X. We use a simple batchwise iterative procedure to generate the initialization of the dictionary for Algorithm 4.

The initialization procedure is listed in Algorithm (5), where OMP is Orthogonal Mathing Pursuit, and OMP $(\mathrm{vec}(Y), I_p \otimes A, N)$ treats the block of samples as a whole compared to the sample-by-sample way in K-SVD. The input of the algorithm (5) includes the total number of non-zeros N in the representation X of a batch of samples Y. There is no guarantee that the Dictionary Approximation algorithm converges, but empirically it provides a good initialization for our Batch-SVD algorithm.

8 Experiments

In this section, we compare our proposed approach to state-of-the-art dictionary learning algorithms on real world data sets, including natural image patches and general machine learning sets. The focus of the experiments is data compression. Data compression is a critical application for dictionary learning. Particularly in the big data regime, if the samples are represented using only a few coefficients, great storage space can be saved. It may also be used in signal communication, where the sender and receiver keep a copy of the dictionary, and only the sparse coefficients are transmitted. In the experiment, we choose a dictionary A and a sparse coefficient matrix X, and try to minimize the reconstruction error $\|Y - AX\|_f$ with a given number of non-zeros in X. The number of non-zeros is set the same as that produced by K-SVD and online dictionary learning, and we compare the reconstruction errors.

8.1 Data Preparation

We use 10 data sets in our experiments. The first one is the demonstration image set provided in the K-SVD toolbox [14], with 5 images: Barbara, boat, house, Lenna, and peppers. For each image we randomly sample 3000 overlapping patches of size 8-by-8 as the training set, and use another randomly sampled 3000 patches as the testing samples in the open-set evaluation. The second data set is the Notre Dame Image library which contains 715 images taken from the Notre Dame Cathedral in Paris. To make the scales consistent, we resize each image to 512-by-512, and then randomly sample $10,000$ patches as the training set. In the testing stage, we randomly sample 3000 image patches from the image library for a total of 100 runs, and report the mean and standard deviation of the reconstruction error. We also carry out experiments on 8 UCI data sets, including mnist, iris, yeast, glass, wine, ecoli, liver-disorder, and heart-disease[2].

8.2 Demonstration of the Dictionary

For a better illustration we train a square dictionary using $10,000$ randomly sampled image patches of 16-by-16 from the Notre Dame Image data set, and thus the dictionary A has dimensions 256-by-256. The average number of non-zeros per sample is $\|X\|_0/p \approx 2.0223$, and $n = 256$. For the MNIST set, we randomly sample 3000 images from the training set, and learn a 784-by-100 dictionary. The average number of non-zeros per sample is around 6.

8.3 Reconstruction on Natural Image Patches

We compare our algorithm with the online dictionary learning, K-SVD, and the overcomplete wavelets with orthogonal matching pursuit. Also the result using a Gaussian random dictionary with OMP is presented as a baseline. For the online dictionary learning, we set $\lambda = 10$. [3] Since the sparsity for the online dictionary learning is only

[2] We remove the sample columns with 'nan' entries in the heart-disease data set.

[3] We use SPAMS [9] with the default batch size 512 in our evaluation.

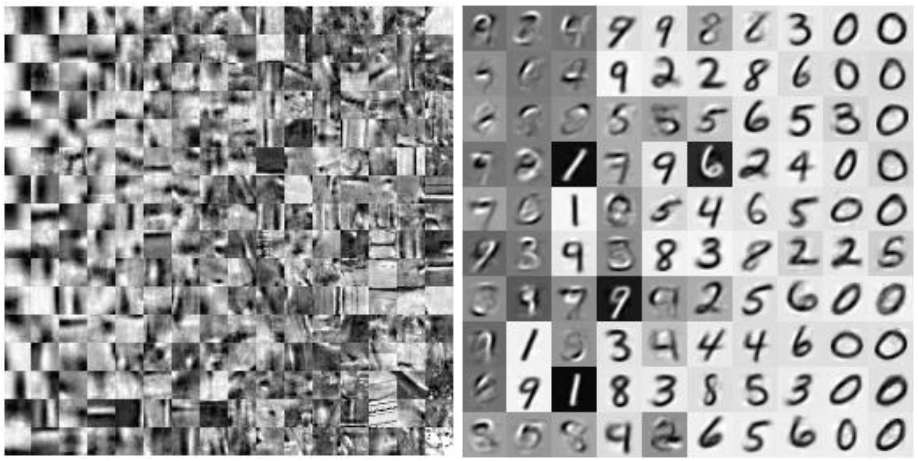

Figure 1: Demonstration of dictionary atoms learned using our algorithm. The left is the learned dictionary from random patches in the Notre Dame library, and the right is from MNIST digit set.

Table 1: Reconstruction Errors on Natural Image Patches. The digit outside the bracket is the average L_2 norm of the errors per patch, and the digit inside the bracket is the standard deviation.

(a) Comparison with K-SVD

$\times 10$	**online**	**KSVD**	**Wvlet**	**Rnd**	**Batch**
barbara	2.9(0.8)	2.0(1.0)	3.1(2.5)	7.4(5.7)	1.8(0.4)
boat	3.2(0.6)	2.1(0.6)	3.0(1.9)	6.4(5.6)	1.9(0.3)
house	2.2(0.8)	1.7(0.9)	3.1(2.7)	6.8(8.3)	1.5(0.4)
lena	2.5(0.6)	1.8(0.7)	2.7(2.0)	6.1(6.2)	1.7(0.3)
peppers	2.9(0.6)	2.1(0.7)	3.0(1.8)	6.2(6.7)	2.0(0.3)
ND	4.3(2.4)	3.1(2.6)	4.0(3.8)	8.2(7.9)	2.7(1.4)

(b) Comparison with Error-based KSVD

ESVD	**KSVD**	**Wvlet**	**Rnd**	**Batch**
2.5(0.4)	3.1(1.7)	5.0(4.1)	10.3(8.1)	2.4(0.6)
2.7(0.3)	3.2(1.1)	4.7(3.3)	8.8(7.8)	2.6(0.4)
2.2(0.8)	2.7(1.9)	5.0(5.1)	8.6(10.4)	2.1(0.8)
2.4(0.4)	3.0(1.8)	5.0(4.9)	8.3(8.7)	2.3(0.6)
2.6(0.3)	2.9(1.1)	4.3(3.4)	7.9(8.6)	2.5(0.5)
2.4(0.8)	2.7(2.2)	3.4(3.2)	7.2(6.6)	2.1(1.0)

Table 2: Reconstruction Errors on the UCI Data Sets. The digit outside the bracket is the average L_2 norm of the errors per sample, and the digit inside the bracket is the standard deviation.

(a) Comparison with K-SVD

$\times 10^{-3}$	**online**	**KSVD**	**Rnd**	**Batch**
liver	22.6(4.0)	7.2(6.9)	51.9(35.6)	5.8(7.3)
iris	20.1(0.1)	522.1(275.9)	102.1(46.3)	11.1(12.4)
yeast	27.8(7.5)	8.1(8.6)	36.2(25.6)	7.9(8.5)
glass	20.1(0.2)	57.6(26.7)	68.5(35.4)	0.9(0.5)
wine	20.1(0.1)	64.8(30.9)	258.1(10.5)	1.6(0.7)
ecoli	27.7(3.5)	8.7(9.5)	50.4(37.3)	2.9(3.7)
heart	21.3(0.9)	173.6(103.5)	345.6(33.8)	8.2(2.7)

(b) Comparison with Error-based KSVD

ESVD	**KSVD**	**Rnd**	**Batch**
0.8(1.9)	7.4(6.5)	40.7(24.3)	0.6(1.3)
491.0(309.7)	331.4(287.1)	152.1(55.4)	2.6(4.1)
2.0(2.9)	14.5(15.0)	37.2(27.4)	3.1(4.9)
50.7(65.4)	45.7(24.3)	442.3(11.7)	3.7(2.3)
80.0(45.7)	66.4(39.6)	797.2(1.5)	4.8(2.8)
1.6(2.4)	16.7(17.9)	41.0(28.7)	2.5(3.5)
172.7(82.4)	182.7(86.0)	334.4(22.6)	5.1(1.6)

softly constrained, we first run the online dictionary learning algorithm and then force $k = \lfloor \|X_{online}\|_0/p \rfloor$ in the K-SVD algorithm, such that the total number of non-zeros in the representation derived using K-SVD is not larger than that of the online learning algorithm. We then set the number of non-zeros in our algorithm to be exactly the same as the K-SVD algorithm. The iteration number of online learning and K-SVD is set as 100. The iteration number for our algorithm is set at 20 with $N_1 = 3$ and $N_2 = 10$. To accelarate the algorithm, the inter-row support switching is only carried out when the objective decrement in the inner-row adjusment is smaller than 0.05. The iteration number of the initialization precedure (5) is set at 80. For the image data sets, the number of atoms in the dictionary A is $n = 256$.

8.4 Reconstruction on UCI Data Sets

We also carried out experiments on the UCI data sets. For all the algorithms, we set the number of atoms in the dictionary $n = 30$. The data vectors are normalized to have unit norm before feeding into the algorithms. Again we first run the online dictionary learning algorithm with $\lambda = 0.02$, and then set $k = \lfloor \|X_{online}\|_0/p \rfloor$, where X_{online} is the coefficient derived using the online learning algorithm. We set the same number of non-zeros for our batch dictionary learning algorithm as that produced by K-SVD. The reconstruction errors are listed in the first part of Table (1) and Table (2). We can

see that the batchwise algorithm works consistently better than the other methods.

8.5 Reconstruction-Error Based K-SVD

We also compared with the reconstruction-error-based K-SVD (ESVD) algorithm proposed in [15]. Since it is not easy to control exactly the number of non-zeros produced by ESVD, again we we first run ESVD and then set the number of non-zeros in our algorithm to be exactly the same as the ESVD algorithm. For comparison the reconstruction errors of the original K-SVD, wavelets, and random dictionary are also presented with $k = \lfloor \|X_{ESVD}\|_0/p \rfloor$. For the Notre Dame library we set the reconstruction error $\epsilon = 30$, yielding an average sparsity $k \approx 9$ per sample. For the UCI data sets we set $\epsilon = 0.01$. The results are presented in the second part of Table (1) and (2), from which we observe that the ESVD algorithm performs reasonably better than the original K-SVD algorithm. The batchwise algorithm, with the same sparsity level, gives better approximations on all the sets except yeast and ecoli.

9 Conclusion

In this paper we propose a monotone dictionary learning algorithm that is optimized for sample batches. The reconstruction error is minimized by a series of support switching procedures withing the sample batch. We prove the objective monotonically decreases and converges in the support switching procedures. Using the proposed block orthogonal matching pursuit algorithm as a warm start, the batchSVD algorithm gives a better approximation in terms of the reconstruction error at the same level of sparsity.

References

[1] B.A.Olshausen and D.J.Field. Emergence of simple-cell receptive eld properties by learning a sparse code for natural images. *Nature*, 1996.

[2] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 2006.

[3] K. Engan, S. Aase, and J. Hakon-Husoy. Method of optimal directions for frame design. In *ICASSP*, volume 5, pages 2443–2446, 1999.

[4] G.Polatkan, M.Zhou, L.Carin, D.Blei, and I.Daubechies. A bayesian nonparametric approach to image super-resolution. *http://arxiv.org/abs/1209.5019*, 2012.

[5] J.Yang, J.Wright, T.Huang, and Y.Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 2010.

[6] K.Skretting and K.Engan. Image compression using learned dictionaries by rls-dla and compared with k-svd. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011.

[7] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. *Advances in Neural Information Processing Systems*, 2006.

[8] M.Aharon, M.Elad, and A.Bruckstein. k-svd: Design of dictionaries for sparse representation. *SPARSE*, 2005.

[9] J. Mairal. *SPArse Modeling Software (SPAMS)*. `http://spams-devel.gforge.inria.fr/index.html`.

[10] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 2010.

[11] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. *Neural Information Processing Systems (NIPS)*, 2008.

[12] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *IEEE Transactions on Image Processing*, 2008.

[13] M.Zhou, H.Chen, J.Paisley, L.Ren, G.Sapiro, and L.Carin. Non-parametric bayesian dictionary learning for sparse image representations. *Neural Information Processing Systems (NIPS)*, 2009.

[14] R.Rubinstein. *K-SVD-Box V13*. `www.cs.technion.ac.il/~ronrubin/software.html`.

[15] R. Rubinstein, M. Zibulevsky, and M. Elad. Effcient implementation of the k-svd algorithm using batch orthogonal matching pursuit. *Technical Report - CS Technion*, 2008.

[16] S.Li, H.Yin, and L.Fang. Group-sparse representation with dictionary learning for medical image denoising and fusion. *IEEE Transactions on Biomedical Engineering*, 2012.

[17] D. Spielman, H. Wang, and J. Wright. Exact recovery of sparsely-used dictionaries. *Conference On Learning Theory*, 2012.